Etat de l'art des stratégies multi-agents de patrouille

Fabrice Lauri
Rapport interne LORIA - INRIA Lorraine

1 Introduction

Selon le dictionnaire Petit Larousse, une patrouille est " une mission de renseignements, de surveillance ou de liaison confiée à une formation militaire (aérienne, terrestre ou navale) ou policière ; désigne également la formation elle-même ".
Selon le dictionnaire Oxford, "patrolling is the act of walking or travelling around an area, at regular interval, in order to protect or to supervise it.".
Une patrouille consiste donc à visiter à intervalle régulier les lieux stratégiques d'une région, dans le but :

- de collecter des renseignements fiables, régulièrement mis à jour par la patrouille,
- de la surveiller en vue de la défendre contre une invasion ennemie,
- ou de faire transiter des informations entre plusieurs lieux d'une région.

Patrouiller efficacement dans un environnement, éventuellement dynamique et bruité, nécessite que :

- le délai entre deux visites d'un même lieu soit minimal,
- dans le cas d'une mission de surveillance, les membres d'une patrouille doivent suivre de préférence des trajectoires non prédictibles par des agents désirant s'infiltrer dans la zone patrouillée.

Les stratégies de patrouille s'adressent à tous les domaines dans lesquels une surveillance ou un contrôle distribué est requis. En particulier, elles peuvent :

- aider les administrateurs à surveiller un réseau contre toute attaque,
- être utilisées par des moteurs de recherche pour indexer les pages Web nouvellement créées ou celles récemment modifiées,
- être employées par des personnages non joueurs (PNJ) dans des jeux vidéos,
- permettre à un groupe de robots d'explorer puis de surveiller continuellement leur environnement

2 Etat de l'Art

2.1 Cadre mathématique

L'ensemble des travaux portant sur les stratégies de patrouille dans les systèmes multi-agents [1, 2, 3, 4, 7, 8, 9] considèrent que l'environnement à patrouiller est connu, bidimensionnel et qu'il peut être réduit à un graphe $G(V, E)$. $V = \{v_1, v_2, \cdots, v_N\}$ représente l'ensemble des N positions (x, y) à visiter dans l'environnement. $E = \{e_{ij}\}$ est l'ensemble des K arcs valués e_ij reliant deux positions v_i et v_j, tels que $e_{ij} = (v_i, v_j, c_{ij})$, une valuation c_{ij} représentant par exemple la distance entre les positions v_i et v_j.

Supposons que l'on dispose de R agents pour patrouiller le graphe G. Le problème de la patrouille peut être envisagé selon deux points de vue. Le premier [1, 2, 3, 4, 7, 8, 9] considère que les agents sont répartis sur les nœuds du graphe et qu'ils doivent les parcourir continuellement de telle sorte que le délai entre deux visites pour chacun des nœuds du graphe soit minimal. Le second point de vue [11, 12] considère que les agents sont disposés sur les arcs du graphe, le but étant alors de traverser les arcs du graphe selon une fréquence quasi-uniforme.

Plusieurs critères peuvent être utilisés afin d'évaluer la qualité d'une stratégie de patrouille. En considérant que les agents sont situés sur les nœuds, les critères suivants devront être calculés au niveau d'un nœud ou au niveau du graphe :

- Critères relatifs à l'Idleness (oisiveté) :
  - Instantaneous Node Idleness (INI) : nombre de pas de temps où un nœudest resté non visité
  - Instantaneous Graph Idleness (IGI) : moyenne de l'Instantaneous Idleness de tous les nœuds pour un instant donné

  - Average Graph Idleness (AvgI) : moyenne des IGI sur n pas de temps
  - Worst Idleness (WI) : plus grande INI apparue pendant les n pas de temps de la simulation

- Exploration Time (ET) : nombre de pas de temps nécessaires aux agents pour visiter au moins une fois tous les nœuds du graphe

Dans le cas où l'on considère que les agents sont disposés sur les arcs, les critères suivants pourront alors être utilisés :

- Cover Time (CT) : nombre de pas de temps nécessaires aux agents pour visiter au moins une fois tous les arcs du graphe (correspond à l'Exploration Time défini ci-dessus mais à propos des arcs)

- Blanket Time (BT) : instant où le ratio entre le nombre de visites entre n'importe lesquels des couples d'arcs est inférieur ou égal à 2. BT apprécie donc le caractère d'uniformité fréquentielle de parcours des arcs.

2.2 Travaux antérieurs

Chevaleyre [3, 4] considère le problème de la patrouille comme un problème d'optimisation combinatoire et s'appuie donc sur les connaissances théoriques accumulées dans les domaines de la recherche opérationnelle et de la théorie des graphes. Il a donné les preuves mathématiques, d'une part, que le problème de la patrouille avec un seul agent est équivalent à une certaine variante du problème du voyageur de commerce (graph-TSP), d'autre part, que ce problème pouvait alors être résolu avec un des algorithmes permettant de déterminer une solution d'une instance du problème du voyageur de commerce (Travelling Salesman Problem). Pour le problème multi-agent de la patrouille, Chevaleyre a réalisé une étude formelle sur plusieurs stratégies possibles de patrouille. Il a entre autres montré qu'en utilisant une " stratégie à cycle unique[1] ", le critère (WI ou AvgI) à minimiser est borné par 3*opt+4*maxijcij. Sur des graphes comportant d'importantes variations de valuations des arcs, cette approche peut donc s'avérer moins intéressante. En

[1] Une stratégie multi-agent est dite à cycle unique si tous les agents parcourent le même cycle dans le même sens et qu'à chaque instant il n'y a pas deux agents qui soient sur le même nœud.

utilisant une " stratégie régionalisée[2] ", le problème devient 15-approximable.

Dans [8, 7], les auteurs ont étudié sept architectures possibles dotant chacun des agents d'un groupe destiné à réaliser des tâches de patrouille. Quatre paramètres entrant en jeu dans l'architecture d'un agent furent explorés : le type d'agent (réactif ou cognitif), le type de communication (aucune, par marques, via un tableau noir ou par messages), la stratégie pour choisir le prochain nœud à visiter et enfin la stratégie de coordination (émergent ou centrale). A propos du choix du prochain nœud, deux aspects furent pris en compte : le type du champs de vision de l'agent (soit local, c'est-à-dire que le prochain nœud est adjacent au nœud courant ; soit global, c'est-à-dire que l'agent perçoit tout le graphe) et l'heuristique de décision : avec l'heuristique basée sur l'Idleness individuelle, l'agent ne considère que ses propres visites tandis qu'avec l'heuristique basée sur l'Idleness partagée, l'agent prend en compte le déplacement des autres agents. Six stratégies de choix du prochain nœud ont ainsi été considérées : LR (locally random), LII (locally individual idleness), LSI (locally shared idleness), GR (globally random), GII (globally individual idleness) et GSI (globally shared idleness).
Dans cet article, seul des graphes comportant des arcs de longueur unitaire furent considérés. Par ailleurs, pour choisir le prochain nœud à visiter, un agent ne prend en compte que le nombre d'arcs le séparant du prochain nœud envisagé. L'Idleness instantanée des nœuds voisins n'est donc pas utilisée dans la prise de décision d'un agent.

Par rapport à [8, 7], les agents dans [1] sont pourvus d'une heuristique qui leur permet de prendre en compte la distance et l'Idleness du prochain nœud. L'heuristique permet ainsi d'éviter de sélectionner comme prochain nœud un nœud qui a l'Idleness la plus grande et qui se situe à une distance importante du nœud où se situe actuellement un agent.

Almeida et al. ont amélioré l'architecture des agents de [8, 7] en concevant dans [2] des agents capables de communiquer à l'aide de messages et de s'entendre mutuellement sur le choix du prochain nœud à visiter. Ces agents peuvent négocier, à l'aide d'un système d'enchères [5] qui permet de faciliter l'entente mutuelle [6]. A l'instant initial, chaque agent reçoit aléatoirement les nœuds qu'il doit visiter. Tout au long de la patrouille, chaque agent tente alors de se constituer un ensemble de nœuds proches les uns des autres, afin

[2]Une stratégie multi-agent régionalisée considère que le graphe est partitionné en régions distinctes et qu'un ou plusieurs agents parcourent les nœuds d'une partition donnée.

de minimiser le délai entre deux visites. Lorsqu'un agent reçoit un nœud qu'il estime ne pas pouvoir visiter dans un temps raisonnable, il initie alors une enchère : il devient demandeur. Les autres agents (alors potentiel offreur) vérifient dans leur ensemble de nœuds s'il existe un nœud éloigné des autres qu'ils pourraient éventuellement échanger avec celui qui est proposé. Si un tel nœud est trouvé, l'agent correspondant fait alors une offre. Une fois toutes les offres déposées, le demandeur les consulte, choisit celle qui est la plus intéressante pour lui et conclut l'affaire avec l'offreur.

Dans [9], Santana et al. ont développé des agents adaptatifs capables d'apprendre à patrouiller dans un environnement à l'aide de techniques issues du domaine de l'apprentissage par renforcement (Reinforcement Learning ou RL). Pour cela, les auteurs ont transposé le problème de la patrouille dans le formalisme des Processus Décisionnels de Markov (MDP). Un espace d'actions et un espace d'états furent définis individuellement pour chaque agent et des modèles de récompense instantanée (modèle d'utilité selfish et modèle d'utilité wonderful life) menant à des performances satisfaisantes à long terme furent développés. L'algorithme Q-Learning [10] fut enfin employé pour réaliser l'apprentissage individuel de chaque agent. Deux architectures ont été considérées : l'architecture Black-Box Learner Agent (BBLA) et l'architecture Gray-Box Learner Agent (GBLA). Les agents de la première architecture ne communiquent pas explicitement entre eux tandis que dans la deuxième architecture, les agents peuvent communiquer leurs intentions au sujet de leurs futures actions. Pour ces deux architectures, l'espace d'actions des agents est constitué d'un nombre d'actions égal au degré du graphe. Pour l'architecture BBLA, l'espace d'états de chaque agent est représenté par :

1) le nœud où l'agent se situe,

2) l'arc d'où l'agent vient,

3) le nœud voisin qui possède l'Idleness la plus grande,

4) le nœud voisin qui possède l'Idleness la plus petite

Pour l'architecture GBLA, l'espace d'états est constitué en plus :

5) des nœuds voisins destinés à être visités par les autres agents

L'espace d'états ainsi défini répertorie environ 10000 et 200000 états possibles, respectivement pour l'architecture BBLA et pour l'architecture

GBLA.

Wagner et al. [11, 12] utilisent des algorithmes inspirés du comportement des fourmis lorsqu'elles explorent un environnement. Dans [12], les agents parcourent le graphe de sommet en sommet, en déposant et " reniflant " des phéromones déposées sur les nœuds. Chaque nœud u du graphe comporte le couple de valeurs (su, tu), où su indique le nombre de marques déposées (en d'autres termes, le nombre de fois où ce nœud a été visité) et tu est le temps où la plus récente marque a été déposée. Initialement, tous les nœuds sont initialisés avec le couple (0,0). Supposons maintenant qu'un agent se trouve sur le sommet u à l'instant t. Pour déterminer vers quel sommet il doit se diriger, l'agent sélectionne parmi tous les sommets voisins de u celui qui a le couple de valeurs le plus petit. Notons v ce sommet. Avant de se diriger vers v, l'agent met à jour s(u) = s(v)+1 et t(u) = t et t est incrémenté. Cette règle est répétée pour chaque agent indéfiniment. Dans [11], les agents parcourent le graphe d'arc en arc en utilisant également des phéromones afin de les guider dans le graphe.

2.3 Résultats théoriques ou expérimentaux

[2] présente une étude expérimentale de plusieurs stratégies de patrouille. Les auteurs ont comparé sept des techniques de patrouille les plus performantes qu'ils ont développé dans leurs équipes, à savoir :

- Conscentious Reactive (CR) et Cognitive Coordinated (CC) [8, 7],
- Heuristic Pathfinder Cognitive Coordinated (HPCC) [1],
- la stratégie à cycle unique (Single Cycle ou SC) [3, 4],
- Gray-Box Learner Agent (GBLA) avec le modèle d'utilité selfish [9],
- Heuristic Pathfinder Two-shots Bidder (HPTB) et Heuristic Pathfinder Mediated Trade Bidder (HPMB) [2].

La comparaison de ces stratégies a porté sur le critère WI. Chaque technique fût évaluée sur six cartes (carte A et B utilisées dans [1, 2, 3, 4, 7, 8, 9], carte avec un itinéraire circulaire, carte avec un couloir, carte avec des îles et carte représentant une grille de nIJuds), en utilisant 15000 pas de temps et une population de 5 ou de 15 agents. Chaque expérience (carte+architecture) fût entreprise 10 fois. Tous les agents furent placés sur le même nœud à

l'instant initial, le nœud de départ variant pour chacune des 10 expériences.

Les résultats montrent que SC obtient les meilleurs résultats (plus petit WI) pour toutes les cartes, sauf pour la carte avec les îles et 15 agents. A l'autre extrême, ce sont les techniques CR et CC qui obtiennent les plus mauvais résultats. Pour toutes les cartes et pour 5 agents, HPCC et GBLA donnent les deuxièmes meilleures performances. Pour 15 agents et pour toutes les cartes, HPCC, HPTB et GBLA fournissent des performances équivalentes.

Dans [11, 12], Wagner et al. montrent que VAW et EAW sont capables de couvrir un environnement en, au plus, n*d et p*n2/k étapes respectivement, n étant le nombre de sommets du graphe, d son diamètre[3], p le degré du graphe et k le nombre d'agents de patrouille.

2.4 Limites des approches

SC est une stratégie de patrouille centralisé, prédéfinie et fixe et en tant que telle, elle sera susceptible de rencontrer des problèmes lorsque l'environnement est dynamique, qu'il comporte plusieurs milliers de nœuds (en raison de la complexité de TSP) et lorsque la tâche de patrouille devra prendre en compte des régions ayant des priorités différentes.

HPCC est plus adaptive que SC, mais elle nécessite un paramétrage pour chaque situation (topologie de la carte et nombre d'agents), afin d'affiner l'importance relative de la distance sur l'Idleness lors du choix du prochain nœud. Comme SC, HPCC risque également de demander une charge processeur importante dans le cas de graphes de grande taille en raison de la complexité de l'algorithme de pathfinding.

GBLA montre une très bonne capacité d'adaptation, même lorsque la population s'accroît. Par ailleurs c'est la seule architecture qui utilise uniquement des informations locales. Néanmoins, le temps nécessaire à la phase d'apprentissage des paramètres (de l'ordre d'une dizaine de millions d'itérations) peut la rendre prohibitive dans le cas de graphes de grande taille.

HPTB et HPMP ne fournissent pas d'aussi bonnes performances que les trois architectures précitées, mais elles ne souffrent pas de leurs limitations.

[3] Le diamètre d'un graphe G, noté $diam(G)$, est la distance maximale rencontrée dans G entre deux couples de sommets.

EAW et VAW sont des techniques très économiques en terme de complexité algorithmique et de place mémoire. Ce sont des techniques totalement distribuées, dans le sens où le même comportement de patrouille est appliqué à chaque agent. Par ailleurs, chaque agent utilise uniquement des informations locales et la communication entre agents, stigmergique, est minimale.

References

[1] A. Almeida. Patrulhamento Multi-Agente em Grafos com Pesos. In *MSc. Dissertation, Universidade Federal de Pernambuco*, 2003.

[2] A. Almeida, G. Ramalho, and *al.* Recent Advances on Multi-Agent Patrolling. In *17th Brazilian Symposium on AI*, 2004.

[3] Y. Chevaleyre. The Patrolling Problem. In *Internal Report of University Paris 9*, 2003.

[4] Y. Chevaleyre. Theoretical Analysis of the Multi-Agent Patrolling Problem. In *International Joint Conference on Intelligent Agent Technology*, 2004.

[5] P. Faratin, C. Sierra, and N.R. Jennings. Using Similarity Criteria to Make Issue Trade-Offs in Automated Negociations. In *Artificial Intelligence Journal*, volume 2, pages 205–237, 2002.

[6] P. Faratin, C. Sierra, and *al.* Negociations Decisions Functions for Autonomous Agents. In *International Journal of Robotics and Autonomous Systems*, volume 24, pages 159–182, 1998.

[7] A. Machado, A. Almeida, and *al.* Multi-Agent Movement Coordination in Patrolling. In *Third International Conference on Computer and Game*, 2002.

[8] A. Machado, G. Ramalho, and *al.* Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In *Proceedings of the 3rd International Workshop on Multi-Agent Based Simulation*, 2002.

[9] H. Santana, G. Ramalho, and *al.* Multi-Agent Patrolling with Reinforcement Learning. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1122–1129, 2004.

[10] R. Sutton and A. Barto. *Reinforcement Learning : An Introduction.* Cambridge, MA, 1998.

[11] I.A. Wagner and *al.* Efficiently Searching a Graph by a Smell-Oriented Vertex Process. In *American Association for Artificial Intelligence*, 1997.

[12] V. Yanowski, I.A. Wagner, and *al.* A Distributed Ant Algorithm for Efficiently Patrolling a Network. In *Algorithmica*, volume 37, pages 165–186, 2003.